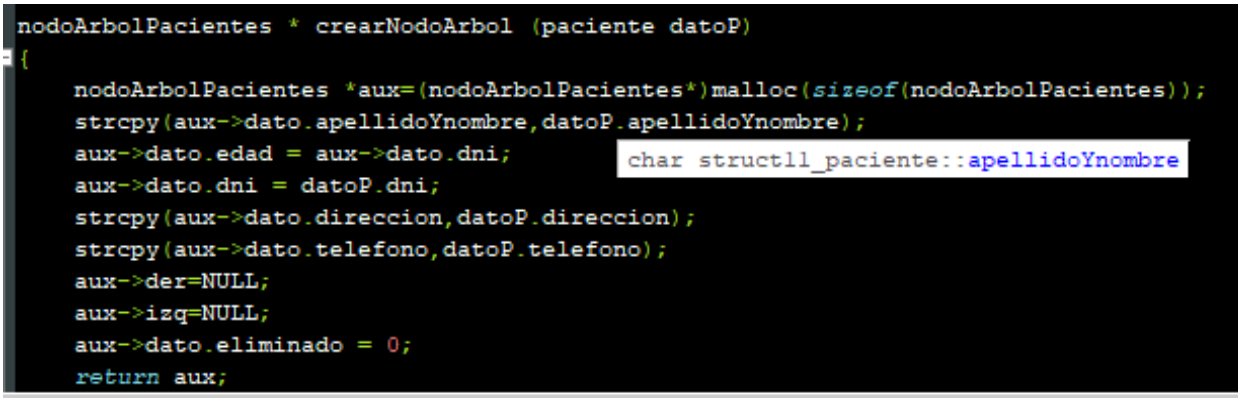
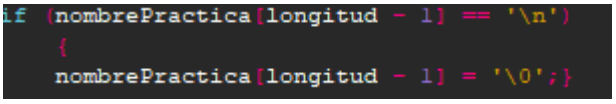
Matriz de soluciones:
Que problema tuvieron y cómo lo resolvieron

- En la función de alta de paciente, la edad no se guarda adecuadamente en el nodo del árbol. Después de revisar todas las funciones de carga y mostrar, se descubrió que el error estaba en que cuando se creaba el nodo del árbol, se le asignaba aux->dato.dni en lugar de datoP.edad a `aux->dato.edad.



- Descubrimos que no nos fijabamos si el paciente existía pero estaba eliminado antes de dar alta, para solucionarlo decidimos agregar en la función alta un if para que en el caso de que el nodo exista pero estaba eliminado se da la opción de agregarlo al sistema otra vez(eliminado=0);

- A la hora de ingresar strings por teclado no los tomaba si había un espacio, se decidió usar fgets para poder tomar todo y que no haya un desborde en el buffer.
- Al ingresar una práctica para agregar a un ingreso y chequear si exquisita devolvió que no existía aunque si lo hacia, asi que se tuvo que crear un código para eliminar los caracteres extra que toma el fgets al ingresar el nombre practico.
-



- En la función para modificar pacientes, al ingresar un DNI que efectivamente estaba en el árbol (aparecía al mostrar el árbol), la función para verificar la existencia del paciente devolvía que no existía. Después de revisar diversas partes del código, como el ingreso, la transferencia de datos de archivo a árbol y de árbol a archivo, descubrimos que nuestra función para verificar si existía el paciente buscaba como si el árbol estuviera ordenado por DNI, cuando en realidad estaba ordenado por el nombre de la persona. Esto provocaba que la función de búsqueda sólo revisara un lado del árbol sin comprobar el otro. El problema se pudo solucionar al rehacer la función de búsqueda para que verificará ambos

lados del árbol y devolviera el nodo en el momento que lo encontrara o NULL si al finalizar de recorrerlo no se encontraba el DNI. Este problema nos hizo darnos cuenta de que al tratar de reubicar el nodo en el caso de que se modificara el nombre, al usar la función de eliminar nodo, se debía buscar por nombre y no por DNI, como se había hecho inicialmente.

```
531     nodoArbolPacientes* existePaciente(nodoArbolPacientes* pacientes, int dniPaciente)
532 -   {
533 -       if (pacientes == NULL)
534 -       {
535 -           printf(".%i. dnifinal\n",dniPaciente);
536             return NULL;
537 -       }
538
539 -       if (dniPaciente == pacientes->dato.dni)
540 -       {
541 -           printf("\n.%i igual. \n",pacientes->dato.dni);
542             return pacientes;
543         }
544 -       else if (dniPaciente < pacientes->dato.dni)
545 -       {

546 -           printf("\n.%imayor. \n",pacientes->dato.dni);
547 -           return existePaciente(pacientes->izq, dniPaciente);
548         }
549 -       else
550 -       {

551 -           printf("\n.%i. \n",pacientes->dato.dni);
552 -           return existePaciente(pacientes->der, dniPaciente);
553         }
```

- En el momento de agregar las funciones al switch, se descubrió que al dar de alta a un empleado, la función se rompía en el área donde se encuentra la línea listaEmpleados = agregarEnOrdenEmpleados(listaEmpleados, nuevo); . Se asumió que la función agregarEnOrden funcionaba correctamente, ya que había sido utilizada previamente y parecía correcta. Se revisaron los datos ingresados para asegurarse de que no hubiera errores en la entrada. Después de varios intentos, se descubrió que la función no funcionaba correctamente cuando el nombre ingresado era mayor que el último nombre de la lista.

  Se volvió a revisar la función agregarEnOrdenEmpleados y se identificó un error al agregar empleados con nombres que, alfabéticamente, deberían ir después del último nombre presente en la lista. Se modificó la función para que tenga en cuenta esta situación y permita la correcta inserción de empleados en orden alfabético, incluso cuando el nombre es mayor que el último de la lista.

- Cuando se estaban probando las funciones en los switch, nos dimos cuenta de errores como que a la hora de modificar un empleado, si lo que se quiere modificar es el DNI y este ya se encuentra cargado en otro empleado, se cargaba de todas formas cuando no debería hacerlo.
- Error descubierto en el sistema fue que, en la función 'dar de baja paciente', al ingresar el DNI, no se elimina al paciente correspondiente. Además, cuando se ingresaba un DNI que no existía, no se mostraba ningún mensaje de error. Faltaba una declaración condicional para verificar la existencia del DNI ingresado y mostrar un mensaje de error en caso de que no existiera.

- En varios case , al mostrar algo se veía 1 segundo y desaparecía , así que se utilizó system(“pause”); para que se pueda visualizar y después volver al menú del switch.
- Al momento de cargar el teléfono que es tipo char se podía ingresar tanto números como letras , se decidió agregarle un chequeo de longitud y isdigit para que el usuario solo pueda ingresar datos válidos , es decir un número menor a 15.
- Al ingresar el Dni, se podia ingresar un numero mayor a 8 digitos, se realizo un chequeo de cantidad de dígitos ingresado para que el usuario ingrese 8 digitos no más ni menos.

- Al principio del TP, se decidió que el número de ingresos sería por paciente y no en general. Por lo tanto, teníamos varios pacientes con el mismo número de ingreso, lo que generaba confusión al modificar ingresos. Es por eso que se decidió asignar el número de ingreso de manera general. Se creó una función que recorría el árbol y las listas de ingresos buscando el mayor número de ingreso para luego añadirle uno y asignarlo al nuevo ingreso.